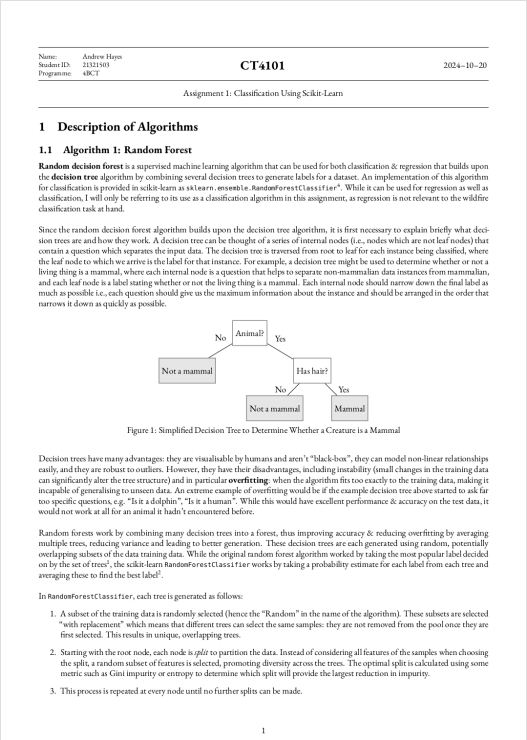
Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Andrew Hayes
Assignment title:	Assignment 1 FINAL PDF Submission Link
Submission title:	CT4101_A1_Hayes_Andrew.pdf
File name:	CT4101_A1_Hayes_Andrew.pdf
File size:	247.33K
Page count:	7
Word count:	3,916
Character count:	20,198
Submission date:	20-Oct-2024 11:38PM (UTC+0100)
Submission ID:	2490764535



Name: Andrew Hayes
Student ID: 21321503
Programme: 4BCT

CT4101 2024-10-20

Assignment 1: Classification Using Scikit-Learn

1 Description of Algorithms

1.1 Algorithm 1: Random Forest

Random decision forest is a supervised machine learning algorithm that can be used for both classification & regression that builds upon the **decision tree** algorithm by combining several decision trees to generate labels for a dataset. An implementation of this algorithm for classification is provided in scikit-learn as `sklearn.ensemble.RandomForestClassifier`[4]. While it can be used for regression as well as classification, I will only be referring to its use as a classification algorithm in this assignment, as regression is not relevant to the wildfire classification task at hand.

Since the random decision forest algorithm builds upon the decision tree algorithm, it is first necessary to explain briefly what decision trees are and how they work. A decision tree can be thought of a series of internal nodes (i.e., nodes which are not leaf nodes) that contain a question which separates the input data. The decision tree is traversed from root to leaf for each instance being classified, where the leaf node to which we arrive is the label for that instance. For example, a decision tree might be used to determine whether or not a living thing is a mammal, where each internal node is a question that helps to separate non-mammalian data instances from mammalian, and each leaf node is a label stating whether or not the living thing is a mammal. Each internal node should narrow down the final label as much as possible i.e., each question should give us the maximum information about the instance and should be arranged in the order that narrows it down as quickly as possible.

Figure 1: Simplified Decision Tree to Determine Whether a Creature is a Mammal

Decision trees have many advantages: they are visualisable by humans and aren't "black-box", they can model non-linear relationships easily, and they are robust to outliers. However, they have their disadvantages, including instability (small changes in the training data can significantly alter the tree structure) and in particular **overfitting**: when the algorithm fits too exactly to the training data, making it incapable of generalising to unseen data. An extreme example of overfitting would be if the example decision tree above started to ask far too specific questions, e.g. "Is it a dolphin", "Is it a human". While this would have excellent performance & accuracy on the test data, it would not work at all for an animal it hadn't encountered before.

Random forests work by combining many decision trees into a forest, thus improving accuracy & reducing overfitting by averaging multiple trees, reducing variance and leading to better generation. These decision trees are each generated using random, potentially overlapping subsets of the data training data. While the original random forest algorithm worked by taking the most popular label decided on by the set of trees[4], the scikit-learn `RandomForestClassifier` works by taking a probability estimate for each label from each tree and averaging these to find the best label[5].

In `RandomForestClassifier`, each tree is generated as follows:

1. A subset of the training data is randomly selected (hence the "Random" in the name of the algorithm). These subsets are selected "with replacement" which means that different trees can select the same samples: they are not removed from the pool once they are first selected. This results in unique, overlapping trees.
2. Starting with the root node, each node is *split* to partition the data. Instead of considering all features of the samples when choosing the split, a random subset of features is selected, promoting diversity across the trees. The optimal split is calculated using some metric such as Gini impurity or entropy to determine which split will provide the largest reduction in impurity.
3. This process is repeated at every node until no further splits can be made.

1